Constraint-based Temporal Reasoning with Preferences

Lina Khatib LINA@EMAIL.ARC.NASA.GOV
Paul Morris PMORRIS@EMAIL.ARC.NASA.GOV
Robert Morris MORRIS@EMAIL.ARC.NASA.GOV
NASA Ames Research Center, Moffett Field, CA 94035 USA

Francesca Rossi FROSSI@MATH.UNIPD.IT
Alessandro Sperduti SPERDUTI@MATH.UNIPD.IT
K. Brent Venable KVENABLE@MATH.UNIPD.IT
*University of Padova, Dept. of Pure and Applied Mathematics,
Via G. B. Belzoni 7, 35131 Padova, Italy*

Abstract

Often we need to work in scenarios where events happen over time and preferences are associated to event distances and durations. Soft temporal constraints allow one to describe in a natural way problems arising in such scenarios.

In general, solving soft temporal problems require exponential time in the worst case, but there are interesting subclasses of problems which are polynomially solvable. In this paper we identify one of such subclasses giving tractability results. Moreover, we describe two solvers for this class of soft temporal problems, and we show some experimental results. The random generator used to build the problems on which tests are performed is also described. We also compare the two solvers highlighting the tradeoff between performance and robustness.

Sometimes, however, temporal local preferences are difficult to set, and it may be easier instead to associate preferences to some complete solutions of the problem. To model everything in a uniform way via local preferences only, and also to take advantage of the existing constraint solvers which exploit only local preferences, we show that machine learning techniques can be useful in this respect. In particular, we present a learning module based on a gradient descent technique which induces local temporal preferences from global ones. We also show the behavior of the learning module on randomly-generated examples.

1. Introduction

Several real world problems involving the manipulation of temporal information can naturally be viewed as having preferences associated with local temporal decisions, where by a local temporal decision we mean one associated with how long a single activity should last, when it should occur, or how it should be ordered with respect to other activities.

For example, an antenna on an earth orbiting satellite such as Landsat 7 must be slewed so that it is pointing at a ground station in order for recorded science or telemetry data to be downlinked to earth. Assume that as part of the daily Landsat 7 scheduling activity a window W is identified within which a slewing activity to one of the ground stations for one of the antennae can begin, and thus there are choices for assigning the start time for this activity. Notice that the time window represents a hard constraint in the sense that no slewing can happen outside such a time interval. Antenna slewing on Landsat 7 has been shown to occasionally cause a slight vibration to the satellite, which in turn might affect the quality of the image taken by the scanning instrument if the scanner is in use during slewing. Consequently, it is preferable for the slewing activity not to overlap any scanning activity. However, since the detrimental effect on image quality occurs only intermittently, this disjointness is best not expressed as a hard constraint. Thus, if there are any start times t within W such that no scanning activity occurs during the slewing activity starting at t, then t is to be preferred. Of course, the cascading effects of the decision to choose t on the scheduling of other satellite activities must be taken into account as well. For example, the selection of t, rather than some earlier start time within W, might result in a smaller overall contact period between the ground station and satellite, which in turn might limit the amount of data that can be downlinked during this period. This may conflict with the preference for attaining maximal contact times with ground stations, if possible.

Reasoning simultaneously with hard temporal constraints and preferences, as illustrated in the example just given, is crucial in many situations. However, in many temporal reasoning problems it is difficult or impossible to specify a preference on each duration. For example, consider a small problem, say with five activities to be scheduled. Ten variables, five for the starting points of the activities and five for the ending points, are needed to model the problem as a temporal constraint satisfaction problem. Preferences on up to fifty different constraints should be given by the user, and it is easy to see how tedious this could be.

Moreover, in real world scheduling problems, it is sometimes easy to see how much a solution is preferred, while it may be virtually impossible to say how specific ordering choices between pairs of events contribute to such a global preference.

This scenario is typical in many cases. For example, it occurs when there is no precise function which describes the assignment of a preference value to a solution. This may happen for example when we just have an expert, whose knowledge is difficult to code as local preferences, but who can immediately recognize a good or a bad solution. Another typical case occurs when the environment in which the solver will work presents some level of uncertainty. In this case, we could have the local preferences, but their effect on a solution could depend on events which are not modeled within the problem. In such a scenario, the available local preferences can be tuned by the ones learned from global preferences which are expressed on complete assignments in which the uncertainty has been revealed.

In this paper we propose to address the scenarios just described, and to tackle the problems they raise, by a combination of temporal reasoning, soft constraints, and machine learning techniques. In particular, the main results of this paper are:

- the definition of a framework, based on temporal constraints (Dechter, Meiri, & Pearl, 1991) and soft constraints (Bistarelli, Montanari, & Rossi, 1995), capable of modeling temporal preferences;
- theoretical complexity results for solving temporal problems with preferences as well as the identification of some tractable sub-classes;
- the design and implementation of two solvers for one of the tractable classes;
- the design and implementation of a learning module capable of eliciting local preferences from global ones;

- a complete experimental scenario.

The paper is organized as follows: Section 2 gives an overview of the background underlying our work. In particular, fundamental definitions and main results are described for temporal constraints, soft constraints, and machine learning. In Section 3 Temporal Constraints with Preferences (TCSPPs) are formally defined and various properties are discussed. After showing that TCSPPs are NP-hard, Simple Temporal Problems with Preferences (STPPs), that is, TCSPPs with one interval on each constraint, are studied. In particular, a subclass of STPPs, characterized by assumptions on both the underlying semiring and the shape of the preference functions, is shown to be tractable. In Section 4 two different solvers for such STPPs are described. Experimental results on the performance of both solvers are supplied in Section 5. In Section 6 a learning module designed for tractable STPPs is described, and experimental results on randomly generated problems are given.

Earlier versions of parts of this paper have appeared in (Khatib, Morris, Morris, & Rossi, 2001; Rossi, Venable, Sperduti, Khatib, Morris, & Morris, 2002b; Rossi, Sperduti, Venable, Khatib, Morris, & Morris, 2002; Rossi, Venable, Sperduti, Khatib, Morris, & Morris, 2002a).

2. Background

In this section we give an overview of the background on which our work is based. First we will describe temporal constraint satisfaction problems (Dechter et al., 1991), a well known framework for handling quantitative time constraints. Then we will define semiring-based soft constraints (Bistarelli, Montanari, & Rossi, 1997). Finally, we will give some background on inductive learning techniques, which we will use in Section 6 for learning local temporal preferences from global ones.

2.1 Temporal constraints

One of the requirements of a temporal reasoning system is its ability to deal with metric information. In other words, a well designed temporal reasoning system must be able to handle information on duration of events ("It will take from ten to twenty minutes to get home") and ordering of events ("Let's go to the cinema before dinner"). Quantitative temporal networks provide a convenient formalism to deal with such information because they consider time points as the variables. A *time point* may be a beginning or an ending point of some event, as well as a neutral point of time. An effective representation of quantitative temporal networks is based on constraints (Dechter et al., 1991).

Definition 1 (TCSP) *A Temporal Constraint Satisfaction Problem (TCSP) consists of a set of variables $\{X_1, \ldots, X_n\}$ and a set of unary and binary constraints over pairs of such variables. The variables have continuous or discrete domains; each variable represents a time point. Each constraint is represented by a set of intervals* [1] $\{I_1, \ldots, I_k\} = \{[a_1, b_1], \ldots, [a_k, b_k]\}$*. A unary constraint T_i restricts the domain of variable X_i to the given set of intervals; that is, it represents the disjunction $(a_1 \leq X_i \leq b_1) \vee \ldots \vee (a_k \leq X_i \leq b_k)$. A*

1. For simplicity, we assume closed intervals; however the same applies to semi-open intervals.

binary constraint T_{ij} over variables X_i and X_j constrains the permissible values for the distance $X_j - X_i$; it represents the disjunction $(a_1 \leq X_j - X_i \leq b_1) \vee \ldots \vee (a_k \leq X_j - X_i \leq b_k)$. Constraints are assumed to be given in the canonical form in which all intervals are pair-wise disjoint.

A TCSP can be represented by a directed *constraint graph* where nodes represent variables and an edge $X_i \longrightarrow X_j$ indicates constraint T_{ij} and it is labeled by the interval set. A special time point X_0 is introduced to represent the "beginning of the world". All times are relative to X_0; thus, we can treat each unary constraint T_i as a binary constraint T_{0i}.

Example 1 *Alice has lunch between noon and 1pm and she wants to go swimming for two hours. She can either go to the pool from 3 to 4 hours before lunch, since she must shower and drive home, or 3 to 4 hours after lunch since it is not safe to swim too soon after a meal. This scenario can be modeled as a TCSP, as shown in Figure 1. There are five variables: X_0, L_s (starting time for lunch), L_e (end time for lunch), S_S (start swimming), S_e (end swimming). The constraint from X_0 to L_s states that lunch must be between 12 and 1pm. The constraint from L_s to L_e states that the duration of lunch must be exactly 1 hour. Similarly for the constraint from S_s to S_e, which states that swimming must last exactly 2 hours. Finally, the constraint from L_s to S_s states that the distance between the start of the swimming activity and the start of lunch must be either between 3 and 4 hours, or between -4 and -3 hours. This means that lunch and swimming can be ordered either way, but in both cases a time between 3 and 4 hours must pass from the start of one activity to the start of the other one.*

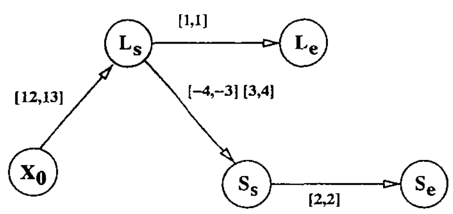

Figure 1: A TCSP.

Given a TCSP, a tuple of values for its variables, say $\{v_1, \ldots, v_n\}$, is called a *solution* if the assignment $\{X_1 = v_1, \ldots, X_n = v_n\}$ does not violate any constraint. A TCSP is said to be *consistent* if it has at least a solution. Also, v_i is a *feasible value* for variable X_i if there exists a solution in which $X_i = v_i$. The set of all feasible values for a variable is called its *minimal domain.* A minimal constraint T_{ij} between X_i and X_j is the set of feasible values for $X_j - X_i$. A TCSP is minimal if its domains and constraints are minimal. It is *decomposable* if every assignment of values to a set of its variables which does not violate the constraints among such variables can be extended to a solution.

Constraint propagation over TCSPs is defined using three binary operations on constraints: union, intersection and composition.

Definition 2 *Let $T = \{I_1, \ldots, I_l\}$ and $S = \{J_1, \ldots, J_m\}$ be two temporal constraints defined on the pair of variables X_i and X_j. Then:*

- *The Union of T and S, denoted $T \cup S$, is*

$$T \cup S = \{I_1, \ldots, I_l, J_1, \ldots, J_m\}.$$

- *The Intersection of T and S, denoted $T \oplus S$, is*

$$T \oplus S = \{K_1, \ldots, K_n\}, \text{ where } K_k = I_i \cap J_j, \text{ for some } i, j.$$

- *The Composition of T and S, denoted by $T \otimes S$ is*

$$T \otimes S = \{K_1, \ldots, K_n\}, K_k = [a + c, b + d], \exists I_i = [a, b], J_j = [c, d].$$

These three operations correspond to the usual operations of union, intersection and composition of constraints (Montanari, 1974). In fact, the union allows only values which satisfy either one of the constraint, while the intersection allows only values which satisfy both constraints. Furthermore, the composition of two temporal constraints, say S and T, defined respectively on the pairs of variables (X_i, X_k) and (X_k, X_j), is a constraint defined on the pair (X_i, X_j) which allows only pairs of values, say (v_i, v_j), for which there exists a value v_k, such that (v_i, v_k) satisfies S and (v_k, v_j) satisfies T.

Given a TCSP, the first interesting problem is to determine its consistency. If the TCSP is consistent, we may wish to find some solutions, or to answer queries concerning the set of all solutions. All these problems are NP-hard (Dechter et al., 1991).

Notions of local consistency may be interesting as well. For example, a TCSP is said to be *path consistent* iff, for each of its constraint, say T_{ij}, we have $T_{ij} \subseteq \oplus_{\forall k}(T_{ik} \otimes T_{kj})$. This notion of local consistency is useful to determine the consistency of a class of TCSPs, which we will define now, in polynomial time.

A TCSP in which all constraints specify a single interval is called a *Simple Temporal Problem*. In such a problem, a constraint between X_i and X_j is represented in the *constraint graph* as an edge $X_i \longrightarrow X_j$ labeled by a single interval $[a_{ij}, b_{ij}]$ that represents the constraint $a_{ij} \leq X_j - X_i \leq b_{ij}$. An STP can also be associated with another directed weighted graph $G_d = (V, E_d)$, called the *distance graph*, which has the same set of nodes as the constraint graph but twice the number of edges: for each binary constraint over variables X_i and X_j, the distance graph has an edge $X_i \longrightarrow X_j$ which is labeled by weight b_{ij}, representing the linear inequality $X_j - X_i \leq b_{ij}$, as well as an edge $X_j \longrightarrow X_i$ which is labeled by weight $-a_{ij}$, representing the linear inequality $X_i - X_j \leq -a_{ij}$.

Each path from X_i to X_j in the distance graph G_d, say through variables $X_{i_0} = X_i, X_{i_1}, X_{i_2}, \ldots, X_{i_k} = X_j$ induces the following *path constraint*: $X_j - X_i \leq \sum_{h=1}^{k} b_{i_{h-1}i_h}$. The intersection of all induced path constraints yields the inequality $X_j - X_i \leq d_{ij}$, where d_{ij} is the length of the shortest path from X_i to X_j, if such a length is defined, i.e., if there are no negative cycles in the distance graph. An STP is consistent if and only if its distance graph has no negative cycles (Shostak, 1981; Leiserson & Saxe, 1988). This means that enforcing path consistency is sufficient for solving STPs (Dechter et al., 1991). It follows that a given STP can be effectively specified by another complete directed graph, called a *d-graph*, where each edge $X_i \longrightarrow X_j$ is labeled by the shortest path length d_{ij} in the distance graph G_d.

In (Dechter et al., 1991) it is shown that any consistent STP is backtrack-free (that is, decomposable) relative to the constraints in its *d-graph*. Moreover, the set of temporal constraints of the form $[-d_{ji}, d_{ij}]$ is the *minimal STP* corresponding to the original STP and it is possible to find one of its solutions using a backtrack-free search that simply assigns to each variable any value that satisfies the minimal network constraints compatibly with previous assignments. Two specific solutions (usually called the *latest* and the *earliest* one) are given by $S_L = \{d_{01}, \dots, d_{0n}\}$ and $S_E = \{d_{10}, \dots, d_{n0}\}$, which assign to each variable respectively its latest and earliest possible time (Dechter et al., 1991).

The *d-graph* (and thus the *minimal network*) of an STP can be found by applying Floyd-Warshall's *All-Pairs-Shortest-Path* algorithm (Floyd, 1962) to the distance graph with a complexity of $O(n^3)$ where n is the number of variables. Such an algorithm initializes a $n \times n$ matrix M to the values of the distance graph. That is, element M_{ij} is initialized to the weight of the constraint from variable X_i to variable X_j in the distance graph, i.e., $M_{ij} := b_{ij}$, while element M_{ji} is initialized to $-a_{ij}$. The diagonal elements of the matrix are instead initialized to 0. The main loop of the algorithm updates each element M_{ij} with $min(M_ij, M_{ik} + M_{kj})$ for every k until quiescence. In the end, either some diagonal element M_{ii} is negative, in which case it is possible to conclude that there is a negative cycle and thus the STP is not consistent, or the elements of the matrix contain the minimum distances d_{ij}. Since, given the d-graph, a solution can be found in linear time, the overall complexity of solving an STP is polynomial.

Example 2 *Consider the scenario described in Example 1 with the additional assumption that Alice can only go swimming in the afternoon. Figure 2 shows the constraint graph of the STP which now models the problems, the corresponding distance graph, the minimal network and the earliest and latest solutions.*

2.2 Soft constraints

In the literature there are many formalizations of the concept of *soft constraints* (Schiex, Fargier, & Verfaillie, 1995; Ruttkay, 1994; Regin, Puget, & T.Petit, 2002). Here we refer to the one described in (Bistarelli et al., 1997, 1995), which however can be shown to generalize and express many of the others (Bistarelli et al., 1997; Bistarelli, Fargier, Montanari, Rossi, Schiex, & Verfaillie, 1996).

In a few words, a soft constraint is just a classical constraint where each instantiation of its variables has an associated element (also called a preference) from a partially ordered set. Combining constraints will then have to take into account such additional elements, and thus the formalism has also to provide suitable operations for combination ($\times$) and comparison ($+$) of tuples of preferences and constraints. This is why this formalization is based on the concept of semiring, which is just a set plus two operations.

Definition 3 (semirings and c-semirings) *A semiring is a tuple* $\langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$ *such that:*

- *A is a set and* $\mathbf{0}, \mathbf{1} \in A$*;*

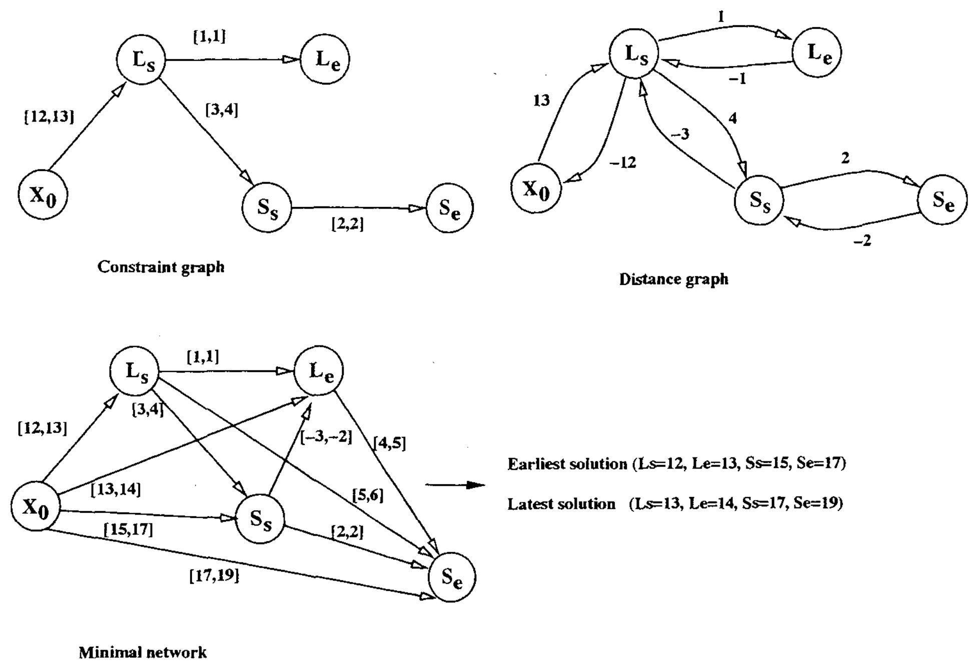
Constraint graph
Distance graph
Minimal network
Earliest solution (Ls=12, Le=13, Ss=15, Se=17)
Latest solution (Ls=13, Le=14, Ss=17, Se=19)

Figure 2: An STP: its constraint graph, distance graph, d-graph (minimal STP), and the earliest and latest solutions.

- $+$ *is commutative, associative and* $\mathbf{0}$ *is its unit element;*
- $\times$ *is associative, distributes over* $+$, $\mathbf{1}$ *is its unit element and* $\mathbf{0}$ *is its absorbing element.*

A c-semiring is a semiring $\langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ *such that:*

- $+$ *is defined over possibly infinite sets of elements of* A *in the following way:*
  - $\forall a \in A$, $\sum(\{a\}) = a$;
  - $\sum(\emptyset) = \mathbf{0}$ *and* $\sum(A) = \mathbf{1}$;
  - $\sum(\bigcup A_i, i \in S) = \sum(\{\sum(A_i), i \in S\})$ *for all sets of indexes* S *(flattening property);*
- $\times$ *is commutative.*

Let us consider the relation $\leq_S$ over A such that $a \leq_S b$ iff $a + b = b$. Then it is possible to prove that (see (Bistarelli et al., 1995)):

- $\leq_S$ is a partial order;
- $+$ and $\times$ are monotone on $\leq_S$;
- $\mathbf{0}$ is its minimum and $\mathbf{1}$ its maximum;

- $\langle A, \leq_S \rangle$ is a complete lattice and, for all $a, b \in A$, $a + b = lub(a, b)$.

Moreover, if $\times$ is idempotent, then $\langle A, \leq_S \rangle$ is a complete distributive lattice and $\times$ is its glb. Informally, the relation $\leq_S$ gives us a way to compare (some of the) tuples of preferences and constraints. In fact, when we have $a \leq_S b$, we will say that *b is better than (or preferred to) a*.

Definition 4 (constraints) *Given a c-semiring $S = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$, a finite set D (the domain of the variables), and an ordered set of variables V, a constraint is a pair $\langle def, con \rangle$ where $con \subseteq V$ and $def : D^{|con|} \rightarrow A$.*

Therefore, a constraint specifies a set of variables (the ones in *con*), and assigns to each tuple of values in D of these variables an element of the semiring set A. This element can be interpreted in many ways: as a level of preference, or as a cost, or as a probability, etc. The correct way to interpret such elements determines the choice of the semiring operations.

Definition 5 (SCSP) *A soft constraint satisfaction problem is a pair $\langle C, con \rangle$ where $con \subseteq V$ and C is a set of constraints over V.*

Note that classical CSPs are isomorphic to SCSPs where the chosen c-semiring is:

$$S_{CSP} = \langle \{false, true\}, \vee, \wedge, false, true \rangle.$$

In fact, since constraints in CSPs are crisp, that is, they either allow a tuple or not, it is possible to model them via a semiring domain with only two elements, say $false$ and $true$: allowed tuples will have associated element $true$ and not allowed ones element $false$. Moreover, constraint combination in CSPs is achieved via a join operation among allowed tuple sets. This can be modeled by choosing *logical and* ($\wedge$) as the multiplicative operator. Finally, to model the projection over some of the variables, as the k-tuples for which there exists a consistent extension to an n-tuple (where n is the total number of variables), it is enough to take the additive operation to be *logical or* ($\vee$).

Fuzzy CSPs (Ruttkay, 1994; Schiex, 1992) extend the notion of classical CSPs by allowing non crisp constraints, that is, constraints which associate a preference level with each tuple of values. Such level is always between 0 and 1, where 1 represents the best value and 0 the worst one. The solution of a fuzzy CSP is then defined as the set of tuples of values (for all the variables) which have the maximal value. The way they associate a preference value with an n-tuple is by minimizing the preferences of all its subtuples. The motivation for such a max-min framework relies on the attempt to maximize the value of the least preferred tuple. It is easy to see that Fuzzy CSPs can be modeled in the SCSP framework by choosing the c-semiring:

$$S_{FCSP} = \langle [0, 1], max, min, 0, 1 \rangle.$$

Definition 6 (combination) *Given two constraints $c_1 = \langle def_1, con_1 \rangle$ and $c_2 = \langle def_2, con_2 \rangle$, their* combination *$c_1 \otimes c_2$ is the constraint $\langle def, con \rangle$, where $con = con_1 \cup con_2$ and $def(t) = def_1(t \downarrow^{con}_{con_1}) \times def_2(t \downarrow^{con}_{con_2})$*[2].

2. By $t \downarrow^X_Y$ we mean the projection of tuple t, which is defined over the set of variables X, over the set of variables $Y \subseteq X$.

The combination operator $\otimes$ can be straightforwardly extended also to finite sets of constraints: when applied to a finite set of constraints C, we will write $\bigotimes C$.

In words, combining constraints means building a new constraint involving all the variables of the original ones, and which associates to each tuple of domain values for such variables a semiring element which is obtained by multiplying the elements associated by the original constraints to the appropriate subtuples.

Constraints can be compared by looking at the semiring values associated to the same tuples. In fact, consider two constraints $c_1 = \langle def_1, con\rangle$ and $c_2 = \langle def_2, con\rangle$, with $|con| = k$. Then $c_1 \sqsubseteq_S c_2$ if for all k-tuples t, $def_1(t) \leq_S def_2(t)$. The relation $\sqsubseteq_S$ is a partial order.

Using the properties of $\times$ and $+$, it is easy to prove that:

- $\otimes$ is associative, commutative, and monotone over $\sqsubseteq_S$;
- if $\times$ is idempotent, $\otimes$ is idempotent as well.

Definition 7 (projection) *Given a constraint $c = \langle def, con\rangle$ and a subset I of V, the* projection *of c over I, written $c \Downarrow_I$, is the constraint $\langle def', con'\rangle$ where $con' = con \cap I$ and $def'(t') = \sum_{t/t\downarrow^{con}_{I\cap con}=t'} def(t)$.*

Informally, projecting means eliminating some variables. This is done by associating to each tuple over the remaining variables a semiring element which is the sum of the elements associated by the original constraint to all the extensions of this tuple over the eliminated variables.

Definition 8 (solution constraint) *The* solution constraint *of an SCSP problem $P = \langle C, con\rangle$ is the constraint $Sol(P) = (\bigotimes C) \Downarrow_{con}$.*

That is, to obtain the solution constraint of an SCSP, we combine all constraints, and then project over the variables in *con*. In this way we get the constraint over *con* which is "induced" by the entire SCSP.

Definition 9 (solution) *Given an SCSP problem P, consider $Sol(P) = \langle def, con\rangle$. A solution of P is a pair $\langle t, v\rangle$ where t is an assignment to all the variables in con and $def(t) = v$.*

Definition 10 (optimal solution) *Given an SCSP problem P, consider $Sol(P) = \langle def, con\rangle$. An optimal solution of P is a pair $\langle t, v\rangle$ such that t is an assignment to all the variables in con, $def(t) = v$, and there is no t', assignment to con, such that $v <_S def(t')$.*

Therefore optimal solutions are solutions which are not dominated by any other solution in terms of preferences. The set of optimal solutions of an SCSP P will be written as $Opt(P)$.

Example 3 *Figure 3 shows an example of a fuzzy CSP. Variables are within circles, and constraints are undirected links among the variables. Each constraint is defined by associating a preference level (in this case between 0 and 1) to each assignment of its variables to values in their domains. Figure 3 shows also two solutions, one of which (S_2) is optimal.*

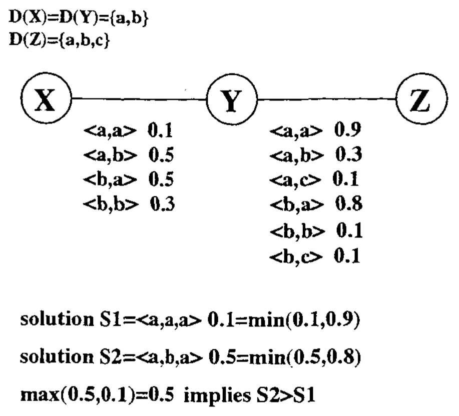

Figure 3: A Fuzzy CSP and two of its solutions, one of which is optimal (S_2).

SCSPs can be solved by extending and adapting the techniques usually used for classical CSPs. For example, to find the best solution, we could employ a branch-and-bound search algorithm (instead of the classical backtracking). Also the so-called constraint propagation techniques, like arc-consistency (Mackworth, 1977) and path-consistency, can be generalized to SCSPs (Bistarelli et al., 1995, 1997).

The detailed formal definition of *constraint propagation* (sometimes called also *local consistency*) for SCSPs can be found in (Bistarelli et al., 1995, 1997). For the purpose of this paper, what is important to say is that a *propagation rule* is a function which, given an SCSP, generates the solution constraint of a subproblem of it. It is possible to show that propagation rules are idempotent, monotone, and intensive functions (over the partial order of problems) which do not change the solution constraint. Given a set of propagation rules, a constraint propagation algorithm applies them in any order until stability. It is possible to prove that constraint propagation algorithms defined in this way have the following properties if the multiplicative operation of the semiring is idempotent: equivalence, termination, and uniqueness of the result.

Thus we can notice that the generalization of local consistency from classical CSPs to SCSPs concerns the fact that, instead of deleting values or tuples of values, obtaining local consistency in SCSPs means changing the semiring value associated to some tuples or domain elements. The change always brings these values towards the worst value of the semiring, that is, the **0**.

2.3 Inductive learning

The problem of learning preferences in STPPs from examples of solutions ratings can be formally described as an inductive learning problem (Russell & Norvig, 2003; Mitchell, 1997). Inductive learning can be defined as the ability of a system to induce the correct structure of a map $t(\cdot)$ which is known only for particular inputs. More formally, defining an example as a pair $(x, t(x))$, the computational task is as follows: given a collection of values, which in turn translates into a training set with many different preference values, helping the module in the inference process.

We conclude by giving some information on the number of iterations and the time used by the algorithm. All the tests have been performed on a machine with a Pentium III 1GHz processor and 512 Mb of RAM. The minimum number of iterations has been 357 while the maximum number has been 3812. The shortest time used has been of 2 minutes and 31 seconds while the longest 8 minutes and 18 seconds. Note that these results were obtained on over 4 different problems since the time needed for a single iteration is not constant.

7. Conclusions and Future Work

We have described a soft constraint based framework for handling temporal preferences. Such a framework is obtained by extending the well known TCSP model for temporal constraint problems with the addition of preference functions which associate each temporal duration or interleaving time with a degree indicating how much it is preferred.

We have given complexity results that show that in general both TCSPs and STPs with preferences are NP-hard. However, we also identified a set of assumptions that guarantee tractability while maintaining a reasonable expressiveness power. We focused our study on such tractable subclass and we proposed two algorithms for finding the optimal solutions of a problem belonging to such a class. One of the algorithms relies on a local consistency procedure (path-solver) while the other one (chop-solver) reduces the problem of finding optimal solutions to testing consistency of problems without preferences.

We have implemented a random generator which has been used to test both algorithms. The tests have shown that chop-solver outperforms path-solver in terms of time needed to solve the instances.

We have also tackled the problem of defining the preference functions on the constraints by examples of solution ratings. Since it is clearly unacceptable to ask the user to define all the preference functions, we consider a scenario in which the user is at least willing to rate some complete schedules. We have applied a machine learning technique in order to induce the local preference functions from such global ratings. We have tested the effectiveness of such a technique which has proved to be quite accurate in predicting the global preference of assignments not rated by the user before.

Many issues remain open. For example, we would like to enhance our framework with the possibility of handling uncertainty deriving from uncontrollable temporal events (some results are in (Rossi, Venable, & Yorke-Smith, 2004)). Another interesting line of work is to consider conditional preferences, that is, preferences that change depending on when other events occur. We also plan to test further our solvers and to try applying different learning techniques for inducing local preferences. We are also considering other optimization criteria and developing specific solvers that follow them, possibly using search.

References

Almeida, L., Langlois, T., Amaral, J., & Plankhov, A. (1998). Parameter adaptation in stochastic optimization. In Saad, D. (Ed.), *Online Learning in Neural Networks*, pp. 111–134. Cambridge University Press.

Biso, A., Rossi, F., & Sperduti, A. (2000). Experimental results on learning soft constraints. In *Proc. Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR 2000)*, pp. 435–444. Morgan Kaufmann.

Bistarelli, S., Fargier, H., Montanari, U., Rossi, F., Schiex, T., & Verfaillie, G. (1996). Semiring-based CSPs and valued CSPs: Basic properties and comparison. *Over-Constrained Systems*, *1106*, 111–150.

Bistarelli, S., Montanari, U., & Rossi, F. (1995). Constraint solving over semirings. In *Proc. Fourteenth International Joint Conference on Artificial Intelligence (IJCAI 95)*, pp. 624–630. Morgan Kaufmann.

Bistarelli, S., Montanari, U., & Rossi, F. (1997). Semiring-based constraint solving and optimization. *Journal of the ACM*, *44*(2), 201–236.

Blythe, J. (2002). Visual exploration and incremental utility elicitation. In *Proc. Eighteenth National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligenc (AAAI/IAAI 2002)*, pp. 526–532. AAAI Press/MIT Press.

Cormen, T., Leiserson, C., & Rivest, R. (1990). *Introduction to Algorithms.* MIT press, Cambridge, MA.

Cousot, P. (1977). Asynchronous iterative methods for solving a fixed point system of monotone equations in a complete lattice. Tech. rep. R. R. 88, Institut National Polytechnique de Grenoble.

Dechter, R., Meiri, I., & Pearl, J. (1991). Temporal constraint networks. *Artif. Intell*, *49*(1-3), 61–95.

Dubois, D., & Prade, H. (1985). A review of fuzzy set aggregation connectives. *Inf. Sci.*, *36*(1-2), 85–121.

Floyd, R. W. (1962). Algorithm 97: Shortest path.. Vol. 36, p. 345.

Goller, C. (1997). A connectionist approach for learning search-control heuristics for automated deduction. Tech. rep., Technical University Munich,Computer Science.

Haykin, S. (1994). *Neural Networks: a comprehensive Foundation.* IEEE Press.

Joachims, T. (2002). Optimizing search engines using clickthrough data. In *Proc. Eighth ACM International Conference on Knowledge Discovery and Data Mining (KDD 2002).* ACM.

Khatib, L., Morris, P., Morris, R. A., & Rossi, F. (2001). Temporal constraint reasoning with preferences. In *Proc. Seventeenth International Joint Conference on Artificial Intelligence (IJCAI 2001)*, pp. 322–327. Morgan Kaufmann.

Leiserson, C. E., & Saxe, J. B. (1988). A mixed-integer linear programming problem which is efficiently solvable. *J. Algorithms*, *9*(1), 114–128.

Mackworth, A. K. (1977). Consistency in networks of relations. *Artif. Intell.*, *8*(1), 99–118.

Mackworth, A. (1992). Constraint satisfaction. In Shapiro, S. C. (Ed.), *Encyclopedia of AI (second edition)*, Vol. 1, pp. 285–293. John Wiley & Sons.

Mitchell, T. (1997). *Machine Learning.* WCB/McGraw-Hill.

Montanari, U. (1974). Networks of constraints: Fundamental properties and applications to picture processing. *Inf. Sci.*, *7*, 95–132.

Pelillo, M., & Refice, M. (1994). Learning compatibility coefficients for relaxation labeling processes. *IEEE Trans. Pattern Anal. Mach. Intell.*, *16*(9), 933–945.

Regin, J.-C., Puget, J., & T.Petit (2002). Representation of soft constraints by hard constraints. In *Proc. of JFPLC'2002.* Université de Nice Sophia-Antipolis.

Rossi, F., & Sperduti, A. (1998). Learning solution preferences in constraint problems. *J. Exp. Theor. Artif. Intell.*, *10*(1), 103–116.

Rossi, F., & Sperduti, A. (2004). Acquiring both constraint and solution preferences in interactive constraint systems. *Constraints*, *9*(4).

Rossi, F., Sperduti, A., Venable, K. B., Khatib, L., Morris, P. H., & Morris, R. A. (2002). Learning and solving soft temporal constraints: An experimental study. In *Proc. Eighth International Conference on Principles and Practice of Constraint Programming (CP 2002)*, No. 2470 in LNCS, pp. 249–263. Springer.

Rossi, F., Venable, K. B., & Yorke-Smith, N. (2004). Controllability of soft temporal constraint problems. In *Proc. Tenth International Conference on Principles and Practice of Constraint Programming (CP 2004)*, No. 3258 in LNCS, pp. 588–603. Springer.

Rossi, F., Venable, K., Sperduti, A., Khatib, L., Morris, P., & Morris, R. (2002a). Solving and learning soft temporal constraints. *AI*IA Notizie*, *4*, 22–26.

Rossi, F., Venable, K., Sperduti, A., Khatib, L., Morris, P., & Morris, R. (2002b). Two solvers for tractable constraints with preferences. In *Proc. AAAI 2002 Workshop on Preferences in AI and CP: Symbolic Approaches.*

Russell, S., & Norvig, P. (2003). *Artificial Intelligence: A Modern Approach.* Prentice Hall.

Ruttkay, Z. (1994). Fuzzy constraint satisfaction. In *Proc. First IEEE Conference on Evolutionary Computing*, pp. 542–547. IEEE.

Saad, D. (1998). *Online Learning in Neural Networks.* Cambridge University Press.

Schiex, T. (1992). Possibilistic Constraint Satisfaction problems or "How to handle soft constraints?". In *Proc. Eighth Annual Conference on Uncertainty in Artificial Intelligence (UAI 1992)*, pp. 268–275. Morgan Kaufmann.

Schiex, T., Fargier, H., & Verfaillie, G. (1995). Valued Constraint Satisfaction Problems: Hard and easy problems. In *Proc. Fourteenth International Joint Conference on Artificial Intelligence (IJCAI 95)*, pp. 631–639. Morgan Kaufmann.

Schwalb, E., & Dechter, R. (1993). Coping with disjunctions in Temporal Constraint Satisfaction Problems. In *Proc. Eleventh National Conference on Artificial Intelligence (AAAI 93)*, pp. 127–132. AAAI Press/MIT Press.

Shostak, R. E. (1981). Deciding linear inequalities by computing loop residues. *J. ACM*, *28*(4), 769–779.

Sutton, R. S. (1992). Adapting bias by gradient descent: An incremental version of delta-bar-delta. In *Proc. Tenth National Conference on Artificial Intelligence (AAAI 92)*, pp. 171–176. MIT Press.

Sutton, R. S., & Whitehead, S. D. (1993). Online learning with random representations. In *Proc. Tenth International Conference on Machine Learning (ICML 1993)*, pp. 314–321. Morgan Kaufmann.

Vila, L., & Godo, L. (1994). On fuzzy temporal constraint networks. *Mathware and Soft computing*, *3*, 315–334.

Zadeh, L. (1975). Calculus of fuzzy restrictions. In *Fuzzy Sets and their Applications*, pp. 1–39. Academic Press.